AFRL-IF-RS-TR-2006-251
Final Technical Report
July 2006

VIRTUAL SECURE DYNAMIC COALITIONS

CACI Technologies, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

APPROVED: /s/

FRANCES A. ROSE
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY Jr., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)*	JUL 06
2. REPORT TYPE	Final
3. DATES COVERED *(From - To)*	Jan 03 – Jun 06
4. TITLE AND SUBTITLE	VIRTUAL SECURE DYNAMIC COALITIONS
5a. CONTRACT NUMBER	F30602-00-D-0221/0032
5b. GRANT NUMBER	
5c. PROGRAM ELEMENT NUMBER	62301E
6. AUTHOR(S)	Francis Lee, John P. Martin
5d. PROJECT NUMBER	Q926
5e. TASK NUMBER	QA
5f. WORK UNIT NUMBER	01
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	CACI Technologies, Inc. 1300 Floyd Avenue Rome New York 13440
8. PERFORMING ORGANIZATION REPORT NUMBER	N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	Air Force Research Laboratory/IFGA 525 Brooks Rd Rome New York 13441-4505
10. SPONSOR/MONITOR'S ACRONYM(S)	
11. SPONSORING/MONITORING AGENCY REPORT NUMBER	AFRL-IF-RS-TR-2006-251
12. DISTRIBUTION AVAILABILITY STATEMENT	*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA#06-527*
13. SUPPLEMENTARY NOTES	

14. ABSTRACT

The goal of the VSDC project was to create a system that could provide secure group communication in a coalition environment. This report documents the design decisions and the outcomes of the Virtual Secure Dynamic Coalitions (VSDC) project. Sections outline the reasons behind the decision to use each of the Defense Advanced Research Projects Agency (DARPA) projects used in the VSDC project.

15. SUBJECT TERMS
Secure group communication in a coalition environment, Virtual Secure Dynamic Coalition

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Frances A. Rose
U	U	U	UL	23	**19b. TELEPONE NUMBER (*Include area code)***

TABLE OF CONTENTS

LIST OF FIGURES

1.0 Project Goal

The goal of the VSDC project was to create a system that could provide secure group communication in a coalition environment. The system needed to be able to support real-time policy enforcement in a constantly changing environment. The system also needed to be able to support policy enforcement in both hardware and software with the ability to completely cut off rogue hosts from the network. In addition all policies used on the network must be able to be centrally managed by policy makers and enforced globally. The operational scenario the project was designed for is a coalition of several countries with varying degrees of trust distributing information of various classification levels. The operation scenario also took into account changing environments where countries or entities within those countries could no longer be trusted and required termination of connectivity from the network. Antigone, Autonomic Distributed Firewall, and the KeyNote systems were used to create this environment.

2.0 Antigone

Antigone is an application-programming interface (API) developed by the University of Michigan to provide a secure group communication middleware layer that can be used by a variety of applications. The Antigone system is built on the notion of a group in the same respect as multicast. A single entity creates a group at run-time. That group can then be joined by other entities after being authenticated. Antigone provides the API for the transport and encryption of data sent to the group. The API also includes facilities for the authentication of entities attempting to join a group.

Antigone was chosen as the initial network authentication and communication system for the VSDC project for its communication, encryption, and authentication capabilities. One of the goals of the VSDC project was to provide a method of communication using groups and limiting access to information based on domain defined prerequisites that prospective members must possess in order to join a group. In addition, the VSDC system needed to be able to dynamically update the policies on both the entities and the groups to simulate changes in the environment. When a change in environment occurs, the groups must maintain integrity and ensure only those users with proper privileges are allowed access to the information being sent to the groups.

The Antigone system allowed for the creation of a series of tools that permitted an entity to create a new group. Once a group has been created the group's creator sets the policy for that group. When another entity, which is not the group's creator, attempts to join a group they are authenticated based on the policy set by the creator. If the connecting entity does not meet the requirements of the policy set by the group's creator they are not allowed access to information for the group.

To protect information sent to groups all communication is encrypted using the OpenSSL library. By using an open standard to provide encryption the Antigone system can provide all of the encryption capabilities supported by the OpenSSL library. The initial key exchanges done by Antigone are done using a form of group diffie-helman exchange.

Antigone uses the Ismene policy language to set policies for the groups. Ismene is a relatively simple policy language using a series of predicates to enforce policy. Each group requires its own policy file to be generated before the group is created. The policy language used by Antigone is composed of simple predicates to validate users. An example of these predicates is shown below:

```
join: timeofday(034500,040000) :: accept;
```

This predicate states that members will only be allowed to join the group between 3:45 and 4:00 AM. At the time of this writing only nine predicates are built into the Antigone system by default. The Ismene language supports extension through custom source code written to support user defined predicates. For a single group several predicates may be defined to further restrict access to the group.

The communication facility provided by Antigone uses standard multicast communication to send data out over the network. This means that every message that is sent to a group will be sent to every member of the group. This also means it is possible for anyone to join the multicast session regardless of the policies implemented and receive the encrypted data. Since the data is encrypted this may or may not be a possible way to attack and defeat the system.

3.0 ADF

The Autonomic Distributed Firewall is a system built for DARPA by Secure Computing and 3Com to provide a hardware firewall solution. The ADF is a network interface card (NIC) that can be either PCI or PCMCIA, and provides encryption and network filtering rules. The system accomplishes this goal by having a VLSI cryptographic engine on the card capable of doing hardware level encryption at a reasonable rate. The card also has memory that is isolated from the host operating system for storing packet filtering rules.

The ADF cards are administered from a single policy server. At the time of this writing only the Microsoft Windows^{TM} operating system is supported. Each of the NICs on the network can have their own filtering rules for packets.

4.0 KeyNote

KeyNote is an API developed by the University of Pennsylvania that provides a generic policy system based on trust management. The trust management system used by KeyNote is composed of several parts: principals, actions, policies, and credentials. Principals are entities authorized to perform some action. Actions are operations with security consequences that are enacted by the system. Policies regulate the actions that principals are allowed to perform. Credentials allow principals to delegate authority to other principals.

KeyNote uses a system of trust established as a hierarchy. A top level “trusted” policy must be used to grant principals a certain level of trust. Trust is granted based on a series of predicates that a principal must comply with. Unlike the Antigone system, the KeyNote system does not use a simple accept or reject mechanism as the only levels of compliance. KeyNote allows multiple user defined levels of compliance to be defined by the policy creators.

Each level of trust granted by the KeyNote system is defined by a policy. An initial top level “trusted” policy is given to the compliance checker specifying those principals that are allowed to grant further trust. This top-level policy must have implicit trust and usually resides on the machine running the compliance checker. The top-level policy also contains some conditions the principals must comply with to grant further trust. These conditionals are based on the actions specified in that principal’s environment. Actions are basically variables with values that are either static or queried when compliance with a policy is verified. Each principle named by the top-level policy can then grant other principals trust with credentials. Credentials are slightly different than the top-level policy since they require the credential creator to sign the policy using their private key. This allows principals to send credentials over insecure networks. KeyNote ensures that no principal can grant trust greater than they are given and that all recipients of a principal’s trust must at a minimum meet the same criteria for actions the principal must meet.

To enforce policy the KeyNote policy checker uses a simple policy system. Each time policy is checked there are three requirements that must be passed into the compliance checker. A trusted policy is used to grant the initial level of trust in the system. A signed credential granting permissions to principals other than those specified in the trusted policy, and an authorizer specifying the principal requesting an action. The credential can make assertions about the environment of the principal requesting the action.

The credential system KeyNote uses contains two key pieces of information for the compliance checker, the principals allowed to request actions, and the conditions that must be met in order for the action to be performed. The conditions are composed of simple boolean conditions that if met return a level of trust based on how many conditions are met. A credential can have multiple levels of trust returned allowing a great deal of flexibility when implementing policy.

The KeyNote system was chosen for the VSDC project for its flexible way of dealing with policy. The VSDC project needed a policy checking system that could handle multiple levels of compliance to deal with DoD standard classification rules. The flexibility offered by using ASCII based policies also allows the ability to build policy strings that can be manipulated dynamically from any program using the system. For the VSDC project, template policy files were used where special keywords marked locations that could be modified at run-time.

5.0 Design Decisions

The design of VSDC had to meet three basic requirements. First was the ability to have runtime policy modification. Situations change quickly and suddenly, often in ways that a static policy would be unable to handle. Relationships between coalition members change in unpredictable ways, so policies governing information exchange must be dynamic to accommodate the changes. The VSDC design addresses this requirement by allowing all policy criteria and user permissions/information to be altered at runtime. Second was the ability to have runtime policy enforcement. Policy implementers must be able to process policy queries and modifications in a reasonable amount of time. While not a strict requirement for VSDC clients, which only enforce local policy, it is vitally important for VSDC servers, which enforce and modify policies for all clients. The VSDC design attempts to minimize the amount of time and resources servers devote to policy enforcement and modification. The third and final requirement was an integration of hardware and software policy. Policy must be able to affect hardware and software with equal ease. True integration of the ADF functionality into the distributed policy management software should result in an environment where a hardware restriction to information is as easy to erect as a software restriction. The VSDC design treats any hardware policy modifications as just another step in software policy enforcement.

The operating system chosen for the VSDC project was Red Hat Linux 9. This was the latest operating system available at the time that supported Antigone, KeyNote, and the ADF NICs. The ADF policy server had to be run on a Microsoft Windows 2000 machine since it didn't support Linux directly and attempts to get the server running under Linux were unsuccessful. The availability of Red Hat and the development tools for Java, C++, and C were the determining factors in the choice of Red Hat. The development language chosen for the project was Java since it is the language the Secure Computing ADF policy server was written in. Java also allowed rapid development of the GUI demonstration applications.

Secure Computing built a system for managing multiple hosts on an internal network with the specific goal of providing encrypted communication with auditing to prevent attacks from inside the network. The system comprised a policy server and NICs built by 3Com. The policy server allows an administrator to set the policy for the entire network from a single interface. The NICs handle the encrypted communication with minimal overhead using hardware encryption that does DES and triple DES encryption. The ADF cards and policy server formed the core the VSDC system would be designed around. Using the ADF cards we gained the ability to encrypt all traffic from host to host without the performance penalty of doing it in software. It also provided the ability to detect packets that violated the policy rules with an audit database. The most important feature of the cards though was the ability to quarantine hosts that might be considered harmful to the network. Since the ADF system provided the capabilities required for VSDC, it was the first component choice and the only system considered for enforcing hardware level policy.

The Antigone API was chosen as the communication medium. It already had the ability to form communication groups over secure multicast. The other option for the part of the VSDC is the integrated KeyNote and Secure Spread system. The KeyNote and Secure Spread system has some limitations that made it unacceptable for use in a real system. One issue is hosts that connect have access to a group for a small period of time before the compliance checking system rejects them if their credentials are insufficient. This would allow arbitrary hosts to attack the system by connecting rapidly to access information. Antigone already had a built in policy language through the use of the Ismene language which allowed simple predicates to be created to enforce access policies to groups.

An additional requirement was added late in the project that required the communication system to be able to query an external data source. While not a necessary requirement to achieve the goal of secure group communication, it required considerable effort to implement policy-based control to an external data source. The KeyNote policy compliance engine was chosen for this function. Its ability to verify generic policy requirements with an extensive policy language made it an ideal candidate for the project. For the back end sample database the Derby database was chosen for its lightweight implementation that supports ANSI SQL.

Figure 1 shows the working network layout with an ADF policy server, Antigone VSDC server, and the clients.

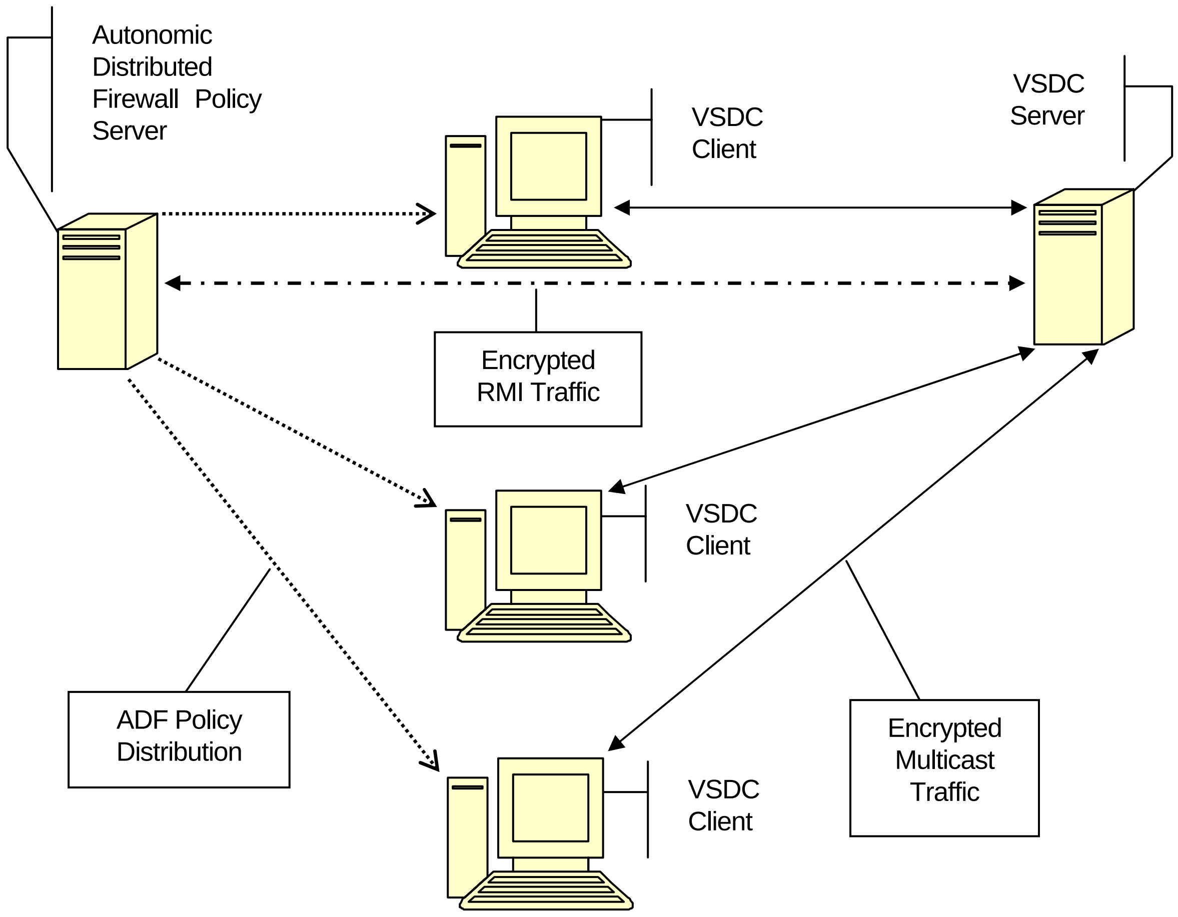


Figure 1: VSDC network architecture

6.0 Implementation

The first step to implementing the VSDC system was getting the ADF hardware installed and working with the Secure Computing policy server. The ADF software was supplied with documentation for installation on Microsoft Windows. The hardware installation guides were supplied for both Microsoft Windows and Linux. The project had four 3Com ADF NICs installed on desktop PCs. No PCMCIA variants of the cards were tested with the project. Both the hardware and software were easily installed.

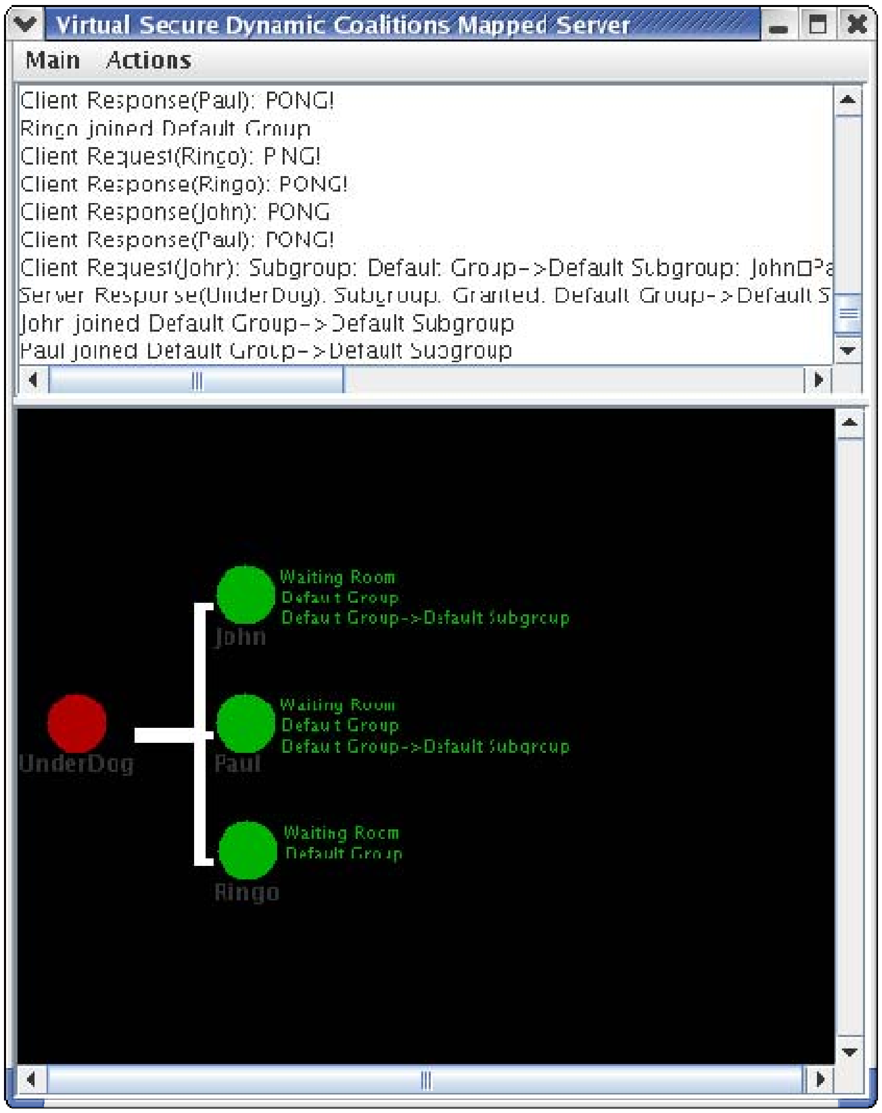
Three of the ADF NICs were installed on machines running Red Hat Linux 9. The fourth was installed on a Microsoft Windows 2000 machine to run the Secure Computing policy server. Since no installation instructions were provided for setting up the server under Linux, after a few attempts Linux was dropped by the project for the policy server. Although the policy server runs using the Java Runtime Environment (JRE), and should theoretically run on any machine capable of running the JRE, the policy server was always problematic.

After the completion of installation and testing that all of the machines could communicate with each other, the next phase was begun. The Antigone API was integrated into the VSDC system. The Antigone API is written in C++, but the policy server is written in Java and depends heavily on Remote Method Invocation (RMI). In order to use Antigone a Java Native Interface (JNI) to Antigone would need to be created. There are limited references on how to create JNI interfaces to C/C++ code available. Initially there were several issues with the Antigone JNI interface including stability problems, linking problems, and issues with native to Java object binding. After the Antigone interface to the policy server was finished, a client/server pair of applications was built to generate traffic and show the communication between the hosts on the network.

7.0 The VSDC system

7.1 VSDC Mapped Server

The “Mapped Server” graphical user interface (GUI) shown in figure 2 below shows three hosts connected to a server. When communication occurs between the hosts and the servers the messages are displayed in the console at the top of the window. The bottom of the window uses indicator lights to display who is communicating. The network topology can be run on any of the machines on the network.


Virtual Secure Dynamic Coalitions Mapped Server
Main
Actions
Waiting Room
Default Group
Default Group->Default Subgroup
John
Waiting Room
Default Group
Default Group->Default Subgroup
Paul
Waiting Room
Default Group
Ringo
UnderDog

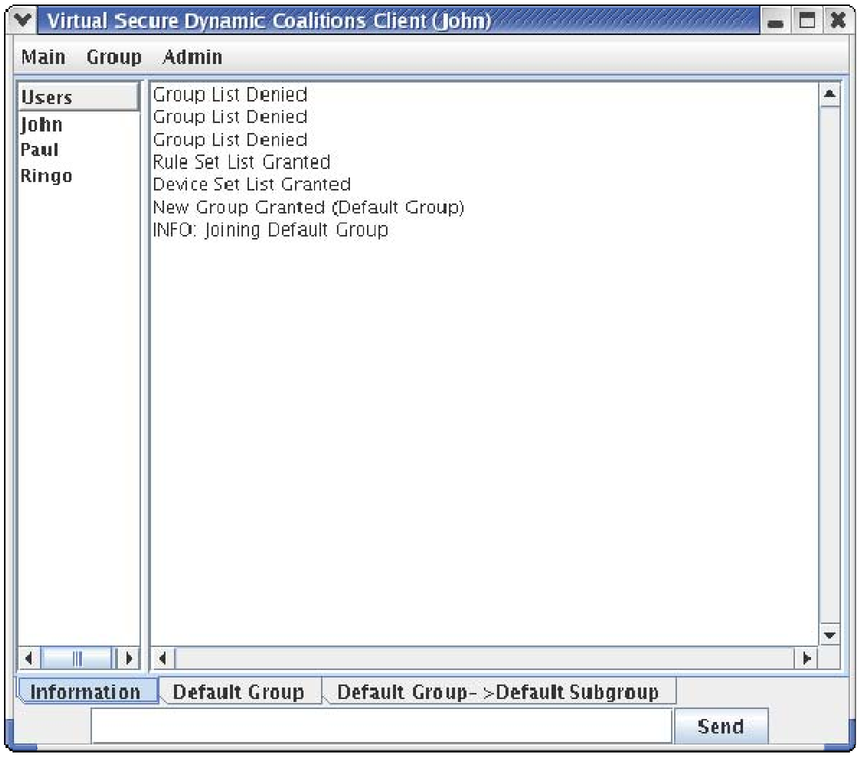
```
Client Response(Paul): PONG!
Ringo joined Default Group
Client Request(Ringo): PING!
Client Response(Ringo): PONG!
Client Response(John): PONG
Client Response(Paul): PONG!
Client Request(John): Subgroup: Default Group->Default Subgroup: John□Pa
Server Response(UnderDog): Subgroup: Granted: Default Group->Default S
John joined Default Group->Default Subgroup
Paul joined Default Group->Default Subgroup
```

Figure 2: Antigone Mapped Server GUI

The "Mapped Server" GUI manages the connections with the clients and also connects the Antigone server to the Autonomic Firewall server in order to manage and distribute the network interface cards' communication policies. The GUI also allows the administrator to "quarantine" a user, causing the Embedded Firewall of a client to stop all traffic on a particular machine.

7.2 VSDC Client

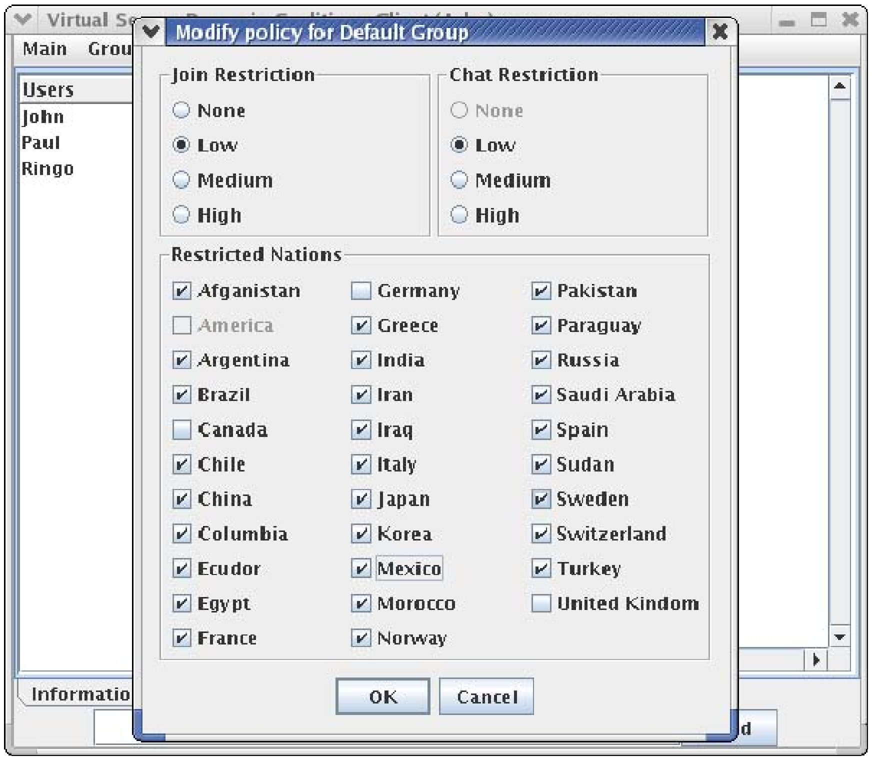
The VSDC Client GUI shown in figure 3 allows users to connect to and utilize the functionalities of the VSDC system.


Virtual Secure Dynamic Coalitions Client (John)
Main
Group
Admin
Users
John
Paul
Ringo
Information
Default Group
Default Group- >Default Subgroup
Send

```
Group List Denied
Group List Denied
Group List Denied
Rule Set List Granted
Device Set List Granted
New Group Granted (Default Group)
INFO: Joining Default Group
```

Figure 3: VSDC Client

The client application allows members of a coalition to create, join, and administer coalition "groups" in a communication chat room. It also allows a group administrator to set up the group policies for joining and participating in the chat room, as shown in figure 4.



Modify policy for Default Group

Join Restriction: None [✓ Low] Medium High

Chat Restriction: None [✓ Low] Medium High

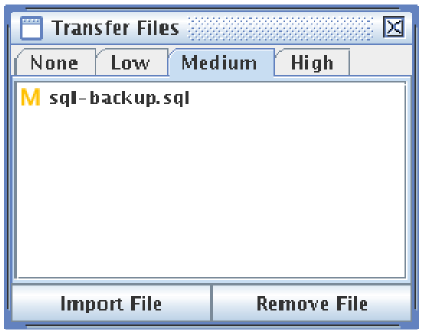
Restricted Nations

[✓ Afganistan]	Germany	[✓ Pakistan]
America	[✓ Greece]	[✓ Paraguay]
[✓ Argentina]	[✓ India]	[✓ Russia]
[✓ Brazil]	[✓ Iran]	[✓ Saudi Arabia]
Canada	[✓ Iraq]	[✓ Spain]
[✓ Chile]	[✓ Italy]	[✓ Sudan]
[✓ China]	[✓ Japan]	[✓ Sweden]
[✓ Columbia]	[✓ Korea]	[✓ Switzerland]
[✓ Ecudor]	[✓ Mexico]	[✓ Turkey]
[✓ Egypt]	[✓ Morocco]	United Kindom
[✓ France]	[✓ Norway]	

Figure 4: Policy settings for a group

8.0 Client File Transfer

The client software also allows for reliable file transfers over Antigone's Secure Spread infrastructure. This allows a client user to send a file simultaneously to all other participating coalition group members securely and robustly. As shown in Figure 5, the file can be readily classified, and the policy enforcement can be implemented for the file transfer to each member.

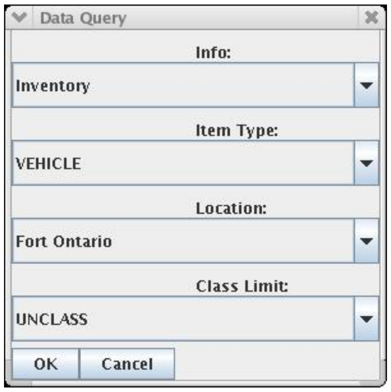


M sql-backup.sql

Figure 5: File transfer in VSDC

9.0 Client Database Querying

The database function was another addition to the Client GUI. Each Client connected to a local Derby SQL engine with different sets of data, and every client had an ability to create a query to retrieve data from all the members of the group. The query builder is shown in figure 6.



Info: Inventory

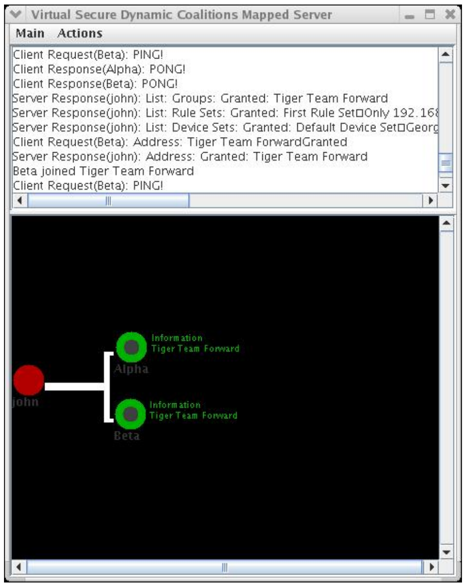
Item Type: VEHICLE

Location: Fort Ontario

Class Limit: UNCLASS

Figure 6: Query builder

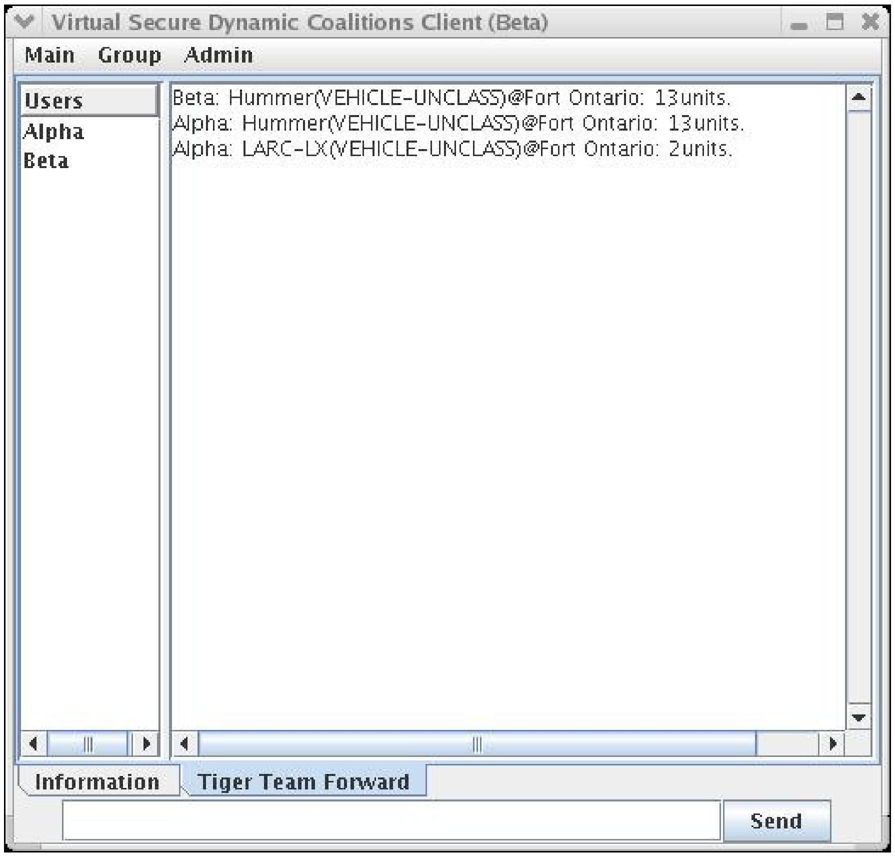
When a member of a group sends out a query, the query would be distributed via the Antigone system shown in figure 7, and each client would query their local database for the information.



```
Client Request(Beta): PING!
Client Response(Alpha): PONG!
Client Response(Beta): PONG!
Server Response(john): List: Groups: Granted: Tiger Team Forward
Server Response(john): List: Rule Sets: Granted: First Rule Set□Only 192.168
Server Response(john): List: Device Sets: Granted: Default Device Set□Georg
Client Request(Beta): Address: Tiger Team ForwardGranted
Server Response(john): Address: Granted: Tiger Team Forward
Beta joined Tiger Team Forward
Client Request(Beta): PING!
```

Figure 7: Active transactions during a query

If the query is valid for client (i.e. database is present, the table is structured correctly, data is present, etc) then the client sends the results via the Antigone system, and the results will be displayed on each client, as illustrated in figure 8.



```
Beta: Hummer(VEHICLE-UNCLASS)@Fort Ontario: 13units.
Alpha: Hummer(VEHICLE-UNCLASS)@Fort Ontario: 13units.
Alpha: LARC-LX(VEHICLE-UNCLASS)@Fort Ontario: 2units.
```

Figure 8: Query results displayed

However, each client has a set of environment variables as shown in figure 9, and a KeyNote policy engine to determine whether or not the received information should or should not be displayed to the user.

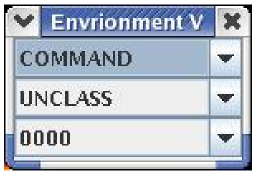


Figure 9: Environment variables of function, classification, and time

10.0 Conclusion

The combination of Antigone and the Autonomic Distributed Firewall network interface cards allowed the development of the Virtual Secure Dynamic Coalition project to complete the following goals:

- Design and develop an information management system that utilizes both hardware and software policies to restrict access.
- Implement the VSDC system by combining previous Defense Advanced Research Projects Agency (DARPA) projects, Antigone and ADF.
- Integrate the software packages with only minimal changes.
- Provide a highly dynamic and flexible policy system.
- Provide the ability to modify user permissions and information at any time.
- Encrypt all multicast User Datagram Protocol (UDP) transmissions used for policy and information distribution.
- State persistence: saves the state of user permissions so that it can be loaded and recreated later.
- Distribute files in a secure manner, to a designated group that is restricted by the policy of the group and the criteria of the sender.
- Allow for database queries for information from all clients in a group.
- Implement KeyNote so that database results may be filtered from Clients that do not have the correct characteristics and environmental variables for accessing the datum.

Although the project was successful in allowing secure, encrypted, policy-based group communication in a controlled environment, there are a few issues, which, if addressed, would pave the way for great improvements in the future of the VSDC approach.

One example of these issues is the current separation of the hardware (ADF) server and the group communication policy (Antigone/Ismene) server. Although they are both Java applications, they both require native libraries that can only run on different operating systems. These libraries provide two different, but related and essential services. And unfortunately, Antigone and KeyNote are written specifically for the Unix/Linux environment, while the ADF server must operate in the Windows environment. Such a division is inefficient and wasteful of resources.

Another major issue is the 3Com EFW PCI NICs for which 3Com has discontinued support; thus making it difficult to implement the ADF on large networks (since the NICs won't be produced). The software is another concern. Antigone, Ismene, and ADF are all unsupported and inactive projects. Without the backing of the original developers, all improvements, upgrades, and bug patches will have to be done by the end user.

Furthermore, Antigone requires that all the client machines operate with their network cards in promiscuous mode. For a user to enable promiscuous mode under Linux, they need to have access to an account that gives administrative permissions on the machine. This is more power than should be allotted to typical users, especially in a paranoid environment.

Currently the policy restrictions are written in Ismene, whose language grammar is outdated and overly restrictive. Greater policy implementation flexibility is required if the policy rulesets are to be more descriptive and complex as demanded by the application.

11.0 Recommendations

The future integration of Matt Blaze’s KeyNote Trust Management System would allow VDSC significantly greater flexible in its policy sets. This added flexibility could allow VSDC’s applications to be expanded beyond the battlefield and into areas such as international diplomacy, urban warfare, or secure intra-governmental group collaboration. Additionally it is recommended that future implementations be made more hardware and platform independent; chief among these changes is the use of both application and hardware-level embedded firewalls. This new architecture would help to avoid situations like the ones we now face (the discontinuation of the EFW NICs) while not compromising the security and integrity required by the application. Another recommendation is to have a way to attach clearance levels to the users. Current policies (for data-filtering/access) use an external, arbitrary clearance level assigner, while in the real world a clearance level is attached to each person before they are allowed access to material or data.

Glossary of Acronyms:

ADF: Autonomic Distributed Firewalls

API: Application Programming Interface

ASCII: America Standard Code for Information Interchange

DARPA: Defense Advanced Research Projects Agency

DES: Data Encryption Standard

DoD: Department of Defense

EFW: Embedded Firewall

GUI: Graphical User Interface

IRC: Internet Relay Chat

JRE: Java Runtime Environment

NIC: Network Interface Card

PC: Personal Computer

PCI: Peripheral Component Interconnect

PCMCIA: Personal Computer Memory Card International Association

SQL: Structured Query Language

UDP: User Datagram Protocol

VLSI: Very Large Scale Integration

VSDC: Virtual Secure Dynamic Coalitions